1 Grammaires non contextuelles

Exercice 1. Donner une grammaire non contextuelle pour chacun des langages suivants (sans trop justifier) :

1. $\mathcal{L}_0 = \{a^i b^j c^k \mid i \neq j \vee j \neq k\}$
2. $\mathcal{L}_1 = \{w \in \{a,b\}^* \mid |w|_a \geq |w|_b\}$
3. $\mathcal{L}_2 = \{w \in \{(,),0,1,*,+\} \mid w$ correspond à une expression arithmétique valide$\}$

Solution 1.

1. On découpe $i \neq j \vee j \neq k$ en $i < j \vee j > i \vee j < k \vee j > k$. Pour On introduit $(x,y) \in \{a,b,c\}^2$ $S_x \to x\ S_x \mid \epsilon$ et $E_{x,y} \to x\ E_{x,y}\ y \mid \epsilon$ qui calculent les termes x^n et $x^n y^n$ la règle est alors : $S \to S_a\ E_{b,c}\ c\ S_c \mid S_a\ b\ S_b\ E_{b,c} \mid S_a\ a\ E_{a,b}\ S_c \mid E_{a,b}\ b\ S_b\ c$
2. $S \to \epsilon \mid aS \mid SaSbS \mid SbSaS$. Montrons cela par récurrence sur la taille des mots. Si un mot est de taille 2 ou moins il est clair qu'il est généré par notre grammaire. Étant donné un mot $w_1 \dots w_n \in \mathcal{L}_\Sigma$ on note $f(k) = |w_1 \dots w_k|_a - |w_1 \dots w_k|_b$.
   S'il existe i tel que $f(i) = 0$ et $1 < i \leq n$ alors on prend le plus petit tel i et on a $w = aw_2 \dots w_{i-1} b w_{i+1} w_n$ ou $w = bw_2 \dots w_{i-1} a w_{i+1} w_n$ avec $w' = w_2 \dots w_{i-1}$ et $w'' = w_{i+1} w_n$ qui vérifient la propriété donc par récurrence $S \xrightarrow{*} w'$ et $S \xrightarrow{*} w''$ soit $S \to SaSbS \mid SbSaS \xrightarrow{*} w$.
   Si un tel i n'existe pas alors on a $f(0) = 0$ et $f(n) > 0$. Soit f est strictement croissante et $w = a^k$ (qui est généré par $S \to aS$) soit on a $1 \leq i < j$ avec $f(i) = f(j)$ et $0 \leq f(j) \leq f(n)$. Notez que $i+1 < j$ car $f(i+1) \neq f(i)$. On coupe alors w en $x = w_1 \dots w_i$, $y = w_{i+2} \dots w_{j-1}$ et $z = w_{j+1} \dots w_n$. Ces trois bouts respectent la propriété car $|x|_a - |x|_b = f(i)$, $|y|_a - |y|_b = f(j) - f(i)$ et $|z|_a - |z|_b$.
   En prenant i et j minimal on a $w_{i+1} \neq w_j$. Dans le cas $w_{i+1} = a$ et $w_j = b$ (resp. $w_{i+1} = b$ et $w_j = a$) comme on a $S \xrightarrow{*} x$, $S \xrightarrow{*} y$, $S \xrightarrow{*} z$ alors $S \to SaSbS \xrightarrow{*} x\ a\ y\ b\ z$ (resp. $S \to SbSaS \xrightarrow{*} x\ b\ y\ a\ z$).
3. $S \to S + S \mid S * S \mid 0 \mid 1 \mid (S)$

Exercice 2. Il y a différentes manières de parser les expressions arithmétiques valides. Trouver une grammaire non ambiguë et dans laquelle les arbres de parsing donne la priorité à * par rapport à + (i.e. 2+3*4 est vue comme l'expression qui vaut 14 et non 20).

Solution 2.

$$\begin{array}{lcr} S & \to & NoPlus \mid NoPlus + S \\ NoPlus & \to & NoPlusTimes * NoPlus \mid NoPlusTimes \\ NoPlusTimes & \to & (S) \mid 0 \mid 1 \end{array}$$

Exercice 3. Prouver que les grammaires définies par les symboles S et $Balanced$ reconnaissent le même langage mais que l'une est ambiguë et pas l'autre :

$$\begin{array}{lcl} Balanced & \to & (OneRight\ Balanced \mid \epsilon \\ OneRight & \to &) \mid (OneRight\ OneRight \end{array} \qquad S \to SS \mid (S) \mid \epsilon$$

Solution 3. Les deux reconnaissent les expressions bien parenthèsées. Pour S c'est assez clair et pour $Begin$ il suffit de remarquer que End reconnaît les expressions e) tels que e est bien parenthèsée.

$S \to SS$ est ambiguë, en effet SSS peut être obtenue comme $[SS]S$ ou $S[SS]$. Tandis que la grammaire introduite par $Begin$ est non ambiguë (comme dérivation est imposée par le premier caractère).

2 Grammaires sensibles au contexte

Exercice 4. Considérons la grammaire suivante :

$$\begin{array}{rcl} S & \to & aSBC \mid aBC \\ CB & \to & BC \\ aB & \to & ab \end{array} \qquad \begin{array}{rcl} bB & \to & bb \\ bC & \to & bc \\ cC & \to & cc \end{array}$$

Quel langage reconnaît-elle ? Justifier.

Solution 4.

Nous allons montrer que $S \xrightarrow{*} w$ implique que w est de la forme (1) $w = a^n b^i c^j \#(B^{n-i}, C^{n-j})$ ou (2) $w = a^n S \#(B^n, C^n)$ pour des n, i, j avec $\#(x, y)$ l'entrelacement de mots x et y.

Nous procédons par récurrence sur la taille d'une dérivation. Pour la taille nulle on a S de la forme (2) ($S = a^0 S \#(B^0, C^0)$) donc c'est bon.

Sinon soit w' respectant notre critère, regardons $S \xrightarrow{*} w' \to w$ on a selon la règle utilisé dans $w' \to w$:

- Si $S \to aSBC$ est la dernière règle alors w' est de la forme (2) et w aussi.
- Si $S \to aBC$ est la dernière règle alors w' est de la forme (2) et w de la forme (1).
- La règle $CB \to BC$ préserve les deux formes.
- Si la règle utilisée est $aB \to ab$ alors on avait w' de la forme (1) avec $w' = a^n b^0 c^0 \#(B^n, C^n)$ et on a $w = a^n b^1 c^0 \#(B^{n-1}, C^n)$.
- Si la règle est $bB \to bb$, $cC \to CC$ ou $bC \to bc$ alors on reste bien dans la forme (1) car un motif cb ne peut pas apparaître.

Finalement, si w est complètement réduit alors w est de la forme $a^n b^n c^n$. Tous les mots de la forme $a^n b^n c^n$ peuvent être obtenus pour $n \geq 1$: on applique $n-1$ la règle $S \to aSBC$ et une fois $S \to aBC$ on obtient $a^n (BC)^n$. On réordonne $(BC)^n$ avec $CB \to BC$ pour obtenir $a^n B^n C^n$ puis en appliquant une fois $aB \to ab$, $n-1$ fois $bB \to bb$, une fois $bC \to bc$ puis $n-1$ fois $cC \to cc$ on obtient bien $a^n b^n c^n$.

Exercice 5. Considérons la grammaire suivante :

$$\begin{array}{rcl} S & \to & CD \\ C & \to & aCA \mid bCB \\ AD & \to & aD \\ BD & \to & bD \\ Aa & \to & aA \end{array} \qquad \begin{array}{rcl} Ab & \to & bA \\ Ba & \to & aB \\ Bb & \to & bB \\ C & \to & \epsilon \\ D & \to & \epsilon \end{array}$$

Quel langage reconnaît-elle ? Justifier.

Solution 5. On va montrer que les dérivations obtenues sont de la forme $wC^?\#(x, y)D^?$ avec $(x, w) \in (\{a, b\}^*)^2$, $y \in \{A, B\}^*$ et $w = x\ rev(lower(y))$ où $X^?$ indique que X peut ou non être présent, rev est la fonction qui retourne un mot et $lower$ transforme les lettres A en a et les B en b.

Pour une dérivation de taille 1 on a $S \to CD$ donc c'est bon.

Pour une dérivation de taille plus grande que 1 selon la dernière règle utilisée on a :

- Les règles $C \to \epsilon$ et $D \to \epsilon$ qui préservent la forme.
- Les règles $C \to aCA$ et $C \to bCB$ préservent aussi la forme. Par exemple pour $C \to aCA$ alors a va être ajoutée à la fin w et A au début de y (et donc a à la fin de $rev(lower(y))$).
- Les règles $Aa \to aA$, $Ba \to aB$, $Ab \to bA$ et $Bb \to bB$ préservent aussi la forme car elles ne ne déplacent pas l'ordre relatif des lettres dans x et y, juste l'entrelacement.
- Enfin les règles $AD \to aD$ et $BD \to bD$ préservent aussi l'ordre car on avait $w = x\ rev(lower(y))$ avec $y = y'A$ donc $x\ rev(lower(y'A)) = x\ rev(lower(y')a) = x\ a\ rev(lower(y'))$ donc $w = (xa)rev(lower(y'))$.

Donc si un mot m avec $S \xrightarrow{*} m$ complètement réduit on a $w \in \{a, b\}^*$ tel que $m = ww$.

Tous les mots de cette forme sont bien accessibles car $S \to wCwD$ pour tout w : c'est évident pour $w = \epsilon$ et par récurrence si $w' = wa$ alors on a $S \xrightarrow{*} wCwD \to waCAwD \xrightarrow{*} waCwAD \to waCwaD$ (on procède de même pour $w' = wb$).

Exercice 6. Considérons la grammaire suivante :

$$S \to aS \mid aSbS \mid \epsilon$$

Quel langage reconnaît-elle ? Justifier.

Solution 6. Nous allons montrer qu'un mot est dans cette grammaire si et seulement si tous ses préfixes contiennent au moins autant de a que de b.

Montrons d'abord par récurrence sur la taille des mots que si un mot est dans cette grammaire alors ses préfixes ont plus de a que de b. La propriété est clairement vraie après 0 dérivation. Et si elle est vraie après n dérivations alors elle est vraie après $n+1$ dérivations car $S \to \epsilon$ ne change pas les a ou b ; $S \to aS$ ne peut qu'ajouter un a et enfin une dérivation $S \to aSbS$ n'ajoute un b à un préfixe que si elle ajoute aussi un a.

Montrons maintenant que tout mot avec notre propriété peut être obtenu par récurrence sur la taille des mots. Soit w un mot respectant la propriété on a $w\epsilon$ et donc $S \to \epsilon$ ou $w = aw'$. Si w' respecte aussi la propriété alors on a $S \xrightarrow{*} w'$ et w peut être obtenu de la façon suivante $S \to aS \xrightarrow{*} aw' = w$.

Maintenant si w' ne respecte pas la propriété, on peut trouver z le plus petit préfixe de w' qui ne respecte pas la propriété. Par mimimalité de z on a $z = x\ b$ avec x respectant la propriété. On a alors $w = a\ x\ b\ y$ avec $a\ x\ b$ qui contient strictement autant de a que de b. Comme w respecte la propriété alors y respecte aussi la propriété (un préfixe p de y correspondant à un préfixe $axbp$ de w). Par récurrence on a donc $S \xrightarrow{*} x$ et $S \xrightarrow{*} y$ donc $S \xrightarrow{*} aSbS \xrightarrow{*} a\ x\ b\ y = w$.

3 Myhill – Nerode

Soit Σ un alphabet et $\mathcal{L}_\Sigma$ un langage dessus (pas nécessairement régulier).

Definition 1. *Soient deux mots $x, y \in \Sigma^{*2}$ on définit la relation suivante :*

$$x \sim_{\mathcal{L}_\Sigma} y \quad \stackrel{\text{def}}{=} \quad \forall z \in \Sigma^* : xz \in \mathcal{L}_\Sigma \Leftrightarrow yz \in \mathcal{L}_\Sigma$$

Exercice 7. Montrer que cette relation est bien une relation d'équivalence (i.e. elle est réfléxive, symétrique et transitive).

Solution 7. C'est la relation d'équivalence induite par la fonction $f(x) = \{z \mid xz \in \mathcal{L}_\Sigma\}$ (i.e. $x \sim_{\mathcal{L}} y \Leftrightarrow f(x) = f(y)$).

Exercice 8. Soient $(x, y) \in \Sigma^{*2}$ et $c \in \Sigma$ montrer que si $x \sim_{\mathcal{L}_\Sigma} y$ alors $xc \sim_{\mathcal{L}_\Sigma} yc$.

Solution 8. Si $x(cz) \in \mathcal{L}_\Sigma$ alors $y(cz) \in \mathcal{L}_\Sigma$ et réciproquement.

Exercice 9. Montrer que si $\sim_{\mathcal{L}_\Sigma}$ a un nombre fini de classes d'équivalences, alors $\mathcal{L}_\Sigma$ est régulier.

Solution 9. Soient $x_1, \ldots, x_n$ des représentants de chacune des classes d'équivalence avec $x_1 \sim_{\mathcal{L}_\Sigma} \epsilon$ et $\mathcal{F} = \{x_i \mid x_i \in \mathcal{L}_\Sigma\}$. D'après l'exercice précédent on peut définir $\delta(x_i, c) = x_j$ tel que $x_i c \sim_{\mathcal{L}_\Sigma} x_j$ et cela ne dépend pas du choix du représentant x_i. On pose donc l'automate $A =< \{x_1, \ldots, x_n\}, \Sigma, \delta, x_1, \mathcal{F} >$ et cet automate reconnaît $\mathcal{L}_\Sigma$. En effet on voit par récurrence sur la taille de w que $\tilde{\delta}(x_1, w) \sim_{\mathcal{L}_\Sigma} w$ et donc $w \in \mathcal{L}_\Sigma \Leftrightarrow \tilde{\delta}(x_1, w) \in \mathcal{L}_\Sigma \Leftrightarrow \tilde{\delta}(x_1, w) \in \mathcal{F}$.

Exercice 10. Montrer que si $\mathcal{L}_\Sigma$ est régulier alors $\sim_{\mathcal{L}_\Sigma}$ a un nombre fini de classes d'équivalences.

Solution 10. Soit $A =< \mathcal{Q}, \Sigma, \delta, i, \mathcal{F} >$ un automate déterministe complet qui reconnaît $\mathcal{L}_\Sigma$. Un mot xz est dans $\mathcal{L}_\Sigma$ si et seulement $\tilde{\delta}(i, xz) \in \mathcal{F}$ et donc ssi $\tilde{\delta}(\tilde{\delta}(i, x), z) \in \mathcal{F}$. Donc pour deux mots x et y tels que $\tilde{\delta}(i, x) = \tilde{\delta}(i, y)$ on a $x \sim_{\mathcal{L}_\Sigma} y$. Si on a n mots de n classes distinctes, on a donc au moins n états dans l'automate or le nombre d'états est fini donc le nombre de classes aussi.

4 Myhill – Nerode : applications

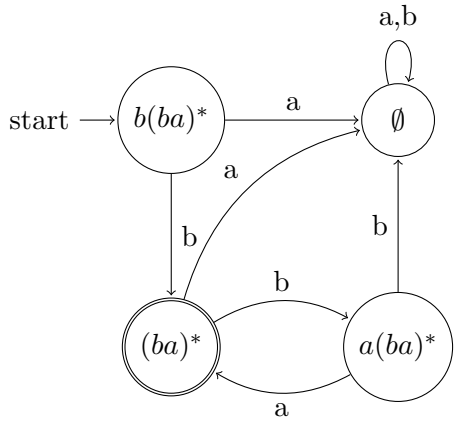
Definition 2. *Calculer les quotients à gauche d'un langage correspond à calculer les classes d'équivalences induites par la relation $\sim_{\mathcal{L}_\Sigma}$ vue plus haut. Les quotients gauches de $\mathcal{L}_\Sigma$ sont les $\mathcal{L}_w$ pour $w \in \Sigma^*$ avec $\mathcal{L}_w = \{x \mid wx \in \mathcal{L}_\Sigma\}$. D'après ce que nous avons vu plus haut les $\mathcal{L}_w$ sont en nombre fini puisque $x \sim_{\mathcal{L}_\Sigma} y \Leftrightarrow \mathcal{L}_x = \mathcal{L}_y$.*

Exercice 11. Trouver les quotients à gauche et l'automate minimal pour chacun des langages suivants sur l'alphabet $\{a, b\}$:

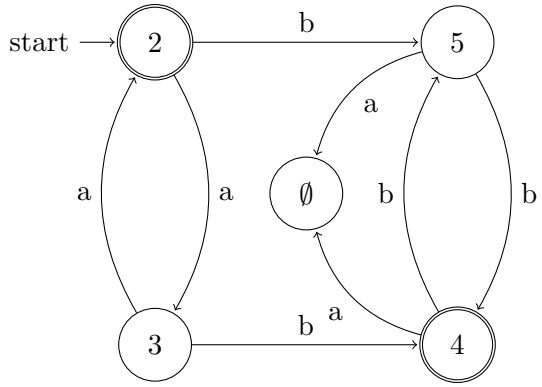
1. $\mathcal{L}_3 = b(ba)^*$
2. $\mathcal{L}_4 = \{a^i b^j \mid i + j \text{ est pair}\}$
3. $\mathcal{L}_5 = \{w \in \{a, b\}^* \mid w \text{ contient exactement une fois le facteur } bb\}$

Solution 11.

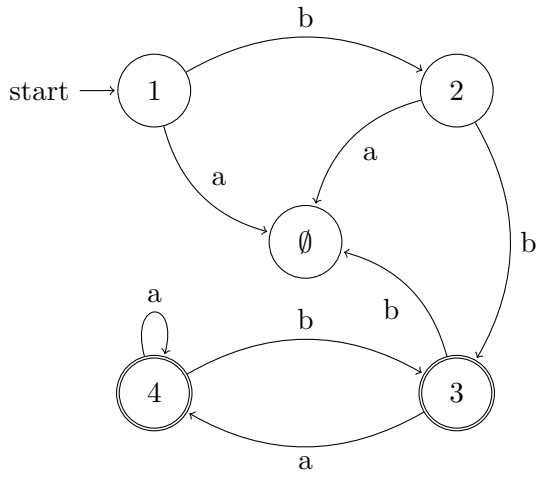
1. Les classes sont $\emptyset$, $b(ba)^*$, $(ba)^*$, $a(ba)^*$ on a donc



2. les classes sont ① $\emptyset$, ② $\{a^i b^j \mid i + j \text{ est pair}\}$, ③ $\{a^i b^j \mid i + j \text{ est impair}\}$, ④ $\{b^j \mid j \text{ est pair}\}$, ⑤ $\{b^j \mid j \text{ est impair}\}$ l'automate est donc :



3. les classes sont ⓪ $\emptyset$, ① $(b?a)^*bb(ab?)^*$, ② $b(ab?)^*|a(b?a)^*bb(ab?)^*$, ③ $(ab?)^*$, ④ $b?(ab?)^*$, l'automate est donc :


start
1
2
∅
4
3
b
a
a
b
b
a
a
b